A Tool For Automatic Verification of Real-Time Expert Systems

B. Traylor, U. Schwuttke, A. **Quan**
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
(81 8) 354-1455
bonnie@puente.jpl.nasa.gov

1.0 introduction

The creation of an automated, user-driven tool for expert system development, validation, and verification is currently ongoing at NASA's Jet Propulsion Laboratory. In the new age of "faster, better, cheaper'' missions, there is an increased willingness to utilize embedded expert systems for encapsulating and preserving mission expertise in systems which combine conventional algorithmic processing and artificial intelligence. The once-questioned role of automation in spacecraft monitoring is now becoming one of increasing importance. In the wake of dwindling budgets, mandated workforce reductions, and increasingly complex systems, there is a growing need for reliable, automated utilities to provide a framework for the development, testing and maintenance of such high-performance systems. Despite the great strides made in this area within the last few years, there are still too few tools which meet the demands of the expanding expert system community, and none which are applicable to the highly specialized one-of-a-kind systems in use at JPL. 'I'his work is an attempt to address some of these concerns,

2.0 **Background**

The expert systems currently in use at JPL are responsible for real-time monitoring and diagnosis over broad domains [Schwuttke, et al. 1994]. These systems are responsible for efficient diagnosis of complex system failures in real-time environments with high data volumes and moderate failure rates. In order to ensure high performance, the algorithmic portions of the systems are implemented in C, while the knowledge-based diagnostic modules arc called upon primarily for decision-making.

The research described in this paper was carried out by the Jet Propulsion laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration.

Requirements for the C modules are obtained via conventional requirements processes and are documented either informally (for advanced development) or in functional requirements documents (for mainstream development), However, the requirements for our expert systems have been obtained in an ad-hoc mode without any semblance of formality. As a result, knowledge transfer takes place through randomly scheduled interviews and ad-hoc phone conversations with end-users based on their limited and occasional availability. The experts involved are spacecraft analysts, and they are a subset of the end-users community. Typically the participation of multiple experts is required, even for knowledge bases of constrained scope. The experts make every effort to provide sufficient insight into diagnosis requirements and knowledge that an expert system developer can implement a rule-base, a little at a time. Weeks or even months might pass between successive conversations, On occasion, an expert system has been considered "finished," only to have end-users request additional development one or two years later, when the original developer of the expert system was no longer available, introducing further inefficiencies into an already problematic development and verification process.

The amount of time that is required to transfer knowledge in this real-world situation probably is several times greater than the amount of time that one or more experts would need to specify the rule base: most of the time required is not dedicated to the specification of rules, but to the transfer and assimilation

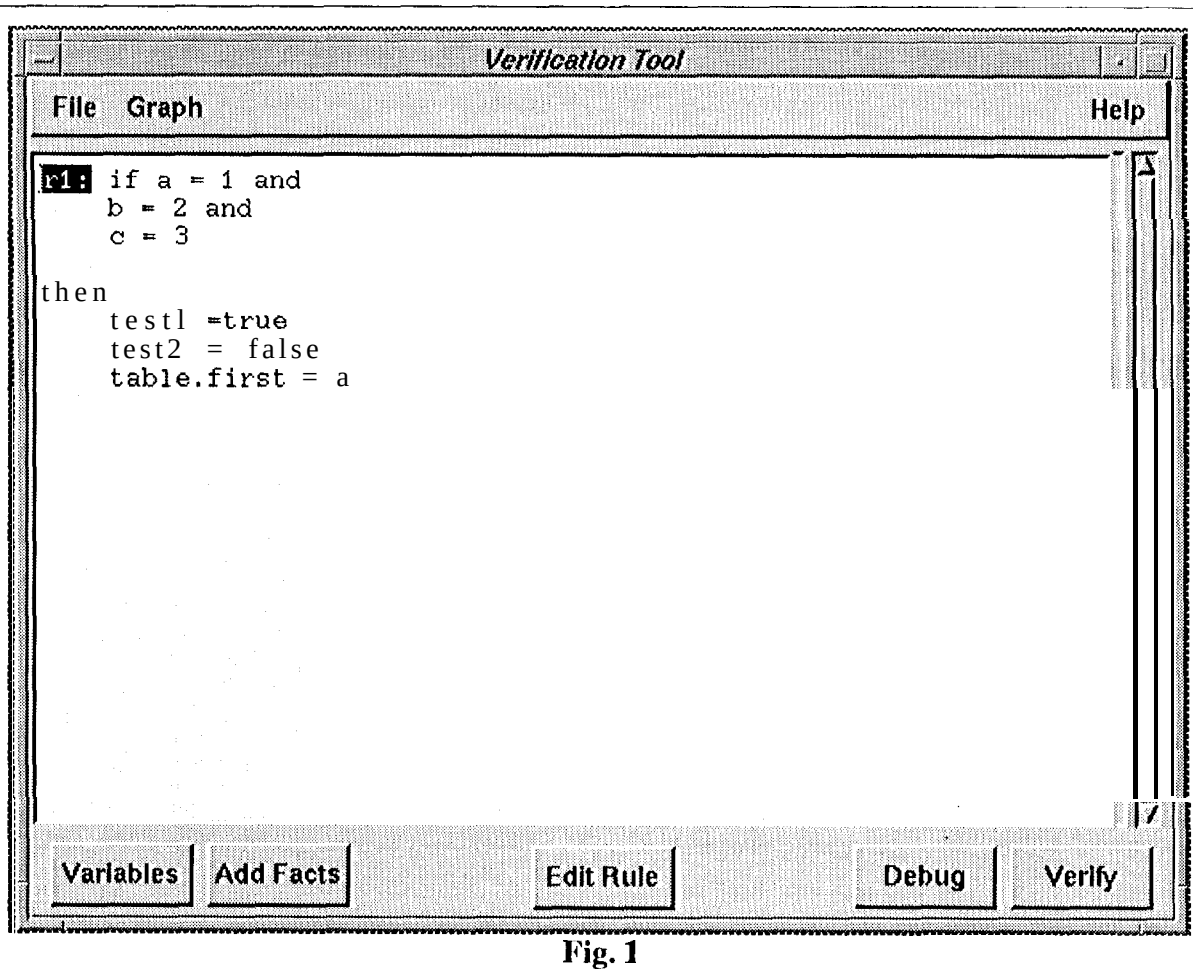

Fig. 1

of domain knowedge needed to implement and adequately test the rulebase. Commercial shells have not been sufficiently simple to use to make this possible, and the expert systems generated by the use of these shells have been sufficiently complex to both understand and verify that multiple simultaneous developers from the end-user community have not been a viable development option.

Therefore, the primary motivations behind the development of this tool are to provide a visual representation of the state of a knowledge base in order to maximize the value of expert time and resources. The resulting tool is intended to provide a means for facilitating the development and future update of knowledge bases. An additional goal has been the creation of a standard platform over which expert systems can be built and maintained, thereby eliminating the need for understanding the collection of shell languages that has become available for various applications. Toward this end, this tool consists of several major components: an update and development tool that utilizes a meta-language and templates for rule definition; a library of standard checkers as commonly discussed in the literature, including a performance monitor; and a visual guide to the rules in the form of a flow graph displaying the relationships among rules, all packaged in a sophisticated graphical user interface.

3.0 Development/Update Component

3.1 Motivating Factors

Clearly, the most difficult phase of development of monitor and analysis systems has been the formalization of expert knowledge into expert system rules. The process of translating rules of thumb and other detailed expert information into highly constrained and precise expert system rules is a tedious and painstaking one at best. Transfer of such knowledge from the expert to the knowledge engineer is often a monumental task, frought with misunderstanding and

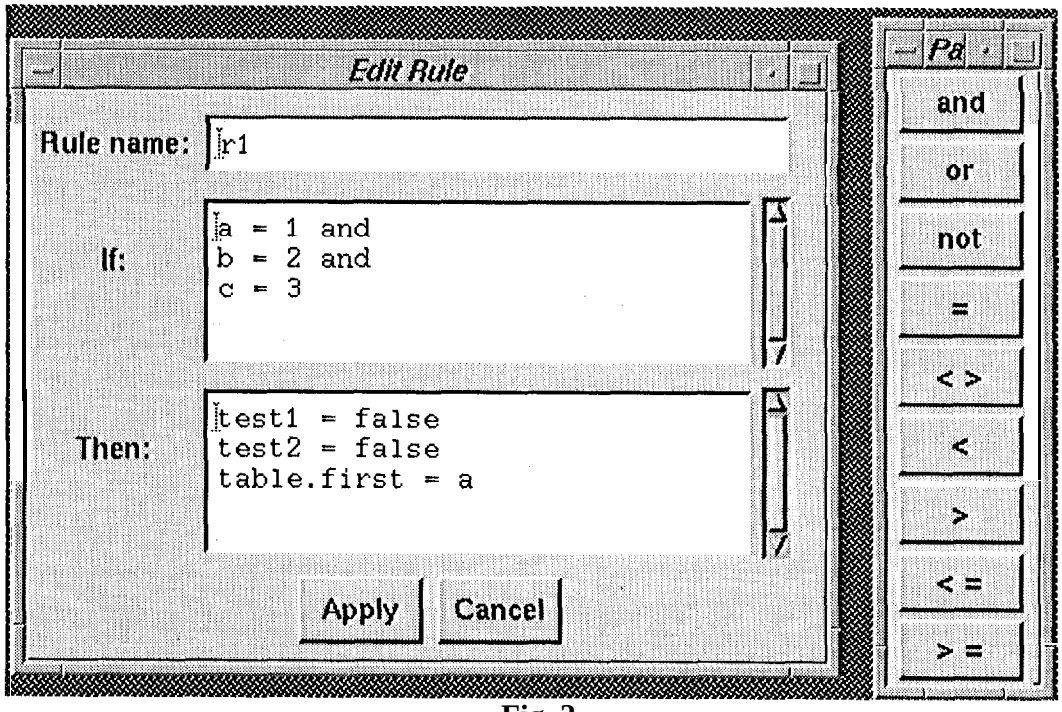

Fig. 2

miscommunication. A possible solution to these problems might be to allow the experts to encode their own rules. However, many shells utilize such complicated syntax and elaborate functionality that experts are understandably hesitant to undertake the task of learning the language in addition to their other responsibilities. In addition, once the knowledge base development is begun, there remains the problem of validation. Although there are several excellent tools available, none is currently compatible with the shell languages currently in use at JPL.

3.2 **Proposed Solution**

Crossing the language barrier is a major challenge in the development of an expert system validation tool in an industry where shell languages are plentiful, and standardization is virtually impossible. A major motivation of this work is to allow the experts to encode knowledge themselves, with only a little outside assistance. This would greatly facilitate the expert system development process, getting expertise online quicker, and with fewer errors due to miscommunication between experts and knowledge engineers. An analysis of rulebases currently in use for monitoring downlink telemetry has yielded some interesting results: the vast majority of the rules in current systems utilize only a fraction of the semantic components of the language, and these components seem to vary only slightly from shell to shell. Accordingly, a meta-language with very simple syntax and few keywords which embodies the commonalities of expert system shell semantics has been created, This language incorporates a very useful intersection of features from several languages, without the overhead of bulky syntax and functionality. Specifically, the meta-language consists primarily of a declaration section for explicit declaration of variables and data structures (the single data structure provided is similar to a C structure or Pascal record), a fact definition section, and a rule definition section. Rules consist of simple statements of the form

if <conditions> ***then*** <actions>

where conditions> are propositional relational tests and <actions> are assignment statements (assertions), print statements and function calls. The structure of the variable declarations, facts, and rules is presented to the user in the form of templates and pushbuttons. Then, as in EVA [Stachowitz et, al., 1987], COVER [Preece, et. al. 1992a], and others, the validation and verification tests can be performed on the meta-language definitions. The verified rulebase can then be translated into the shell language of choice with

minimal effort. The key difference here is that rules are written directly into the meta-language, eliminating the need for a full-scale translation from the shell language into an internal representation.

3.3 Tool Design

In an effort to provide the expert with a hospitable and user-friendly environment, the tool is presented in a sophisticated XWindows/Motif-based graphical user interface (GUI). The top-level window (see Fig. 1) features a scrolled, read-only text window in which the rules are displayed, and which is updated as rules are edited. Pull-down menus are available for file activites, graphical displays, and general and specific help. File activities include loading, saving, and printing the rule base. The "Graph" menu will provide options for either viewing a graphical display of the relationships among the rules, or performance monitoring. The performance monitoring functions will provide the user with various statistics related to the execution of the rulebase, such as frequent firing sequences, most frequently fired rules, etc., as well as providing a color-coded graphical display, highlighting rulebase features with primary colors. Because simplicity and ease of use are primary concerns, the user will enter candidate rules into the rulebase via a template, which will provide the user with the appropriate format of the rule. For example, an existing rule may be changed by first selecting the appropriate rule name with the mouse. This will cause the button labeled "Add Rule" to be changed from its default state and to display the label ' 'Edit Rule. " The user can then click on this button, which will pop up the "Edit Rule" window (see Fig. 2). The three panes here are labeled "Rule Name:," "if" and "then," and these panes will be filled with the information appropriate to the rule which was selected from the top-level text window. Clicking on the "Apply" button will cause this updated rule to be subjected to various syntactic checks, as well as whatever verification tests which are applicable for the incomplete rule base, and also causes updates to be reflected in the top-level window. When no text is selected (the default state) the' 'Add Rule" label will be displayed, and clicking on this button will cause a blank template to be displayed. In addition, an optional palette will accompany the rule-editing window, allowing users to enter frequently-used language elements with the push of a button. The "Add Vars" and "Add Pacts" pushbuttons for editing variable and fact definitions are similar in function. The "Debug" function will provide an interface for watching variables, inserting breaks, and stepping through the rules as they fire. The "Verify" option is discussed in the next section.

4.0 Verification Component

Various issues were considered in the determination of knowledge base tests to include in this tool. The primary motivating factor is that the intended user is not necessarily a knowledge engineer, and therefore, the need for extensive automatic syntax and semantic checking is paramount. Before a candidate rule is accepted for insertion into the rulebase, the rule will undergo tests for possible syntactic and semantic problems such as duplicate rule names, as well as for conditions such as inconsistency, subsumption (redundancy), and looping, among others, before being added to the rulebase. ([Preece et al. 1992b] provides concise, clear definitions of verification tests performed by several existing systems, in addition to providing a framework for comparison,) Syntactic checking, such as that used by CHECK [Nguyen et al. 1985] will provide a core, however, initial analysis indicates that extended semantic checking like that described in [Suwa, et al., 1982] and [Stachowitz, et al., 1987] will also be possible because the domains of many applicable areas at JPL are mutually exclusive, however, this remains an open issue. The tool will also include capabilities for dynamic analysis of the knowledge-base -- primarily for determination of frequently-traversed paths, frequently and infrequently used rules, and other performance meters. A display of graphical elements corresponding to the rules and the relationships among them will also be included in the tool. This is clearly a very effective medium for the presentation of certain types of errors and also for performance monitoring. Although neither the graphical display of the rules, nor the graphical user interface in which the tool will be packaged is of any interest in ES V&V research, they are essential components of a real-world application -- a tool which is too abstruse or esoteric to be used with minimal effort on the part of the expert/user is of 1 ittle value for producing usable systems.

5.0 **Open Issues**

Because the exact specification of this tool is work-in-progress, many design-related issues are yet to be determined, First, it is imperative that as many structural and semantic verification tests are included as possible. Some tests, however, are computationally intensive for larger rule bases, and should therefore be included in the verification utility, rather than being applied automatically before a rule is inserted into the rulebase. An analysis of the complexity of various tests is yet to be completed, Similar complexit y issues related to the exact nature of the graphical display of rules are also still under investigation, and therefore, the precise features of this utility are yet to be determined. One very important feature of this tool is that it should allow the execution of the rule-basc to be monitored after it has been embedded within the calling C code. This feature will be a dramatic improvement over past testing capabilities, however the precise mechanics of this function have not yet been identified. In addition, the possibility of incorporating an extended semantic semantic checker which utilizes meta-knowledge of the domains in question, as mentioned before, is still under consideration,

6.0 **Conclusion**

Due to the inavailability of expert system validation and verification tools which interface with more than one or two shell languages, other approaches to the problem of producing reliable expert systems are clearly necessary. Utilization of a meta-language front-end to multiple expert system shells, combined with powerful verification algorithms and a sophisticated graphical user interface provides a practical solution to an old problem. This tool, however, is in no way designed to be an all-purpose expert system development tool. It is not meant for the development of large, standalone expert systems with extremely many rules, or which require complex language functionality, However, it can be successfully applied to small or medium-sized knowledge bases which require only simple forward-chaining if-then statements. This approach to the design and maintenance of embedded expert systems has several advantages. First, by producing only a front-end to existing expert system shells, rather than producing a full-scale validation and verification system for a single general purpose shell, and by utilizing existing verification techniques, the time required to develop the tool will be very short. Additionally, it allows the experts to be the developers of the expert systems, which in turn eliminates errors caused by miscommunication with knowledge engineers. With the use of expert systems for critical activities on the rise, the development of sophisticated, yet easy-to-use tools for validation and verification is critical.

7.0 **References**

Nguyen, T. A., Perkins, W. A., Laffey, T. J., and Pecora, D. 1985. Checking an Expert Systems Knowledge Base for Consistency and Completeness. In *Proc. 9th Internat'l Joint Conference on Artificial intelligence,* Vol 1,375-378, AAAI.

Preece, A.D., Shinghal, R., and Batarekh, A. 1992. Principles and Practice in Verifying Rule-Based Systems. *The Knowledge Engineering Review,* Vol 7:2, *115-141.*

Preece, A. D., Shinghal, R., and Batarekh, A. 1992. Verifying Expert Systems: a Logical Framework and a Practical Tool. *Expert Systems with Applications 4(2/3).*

Schwuttke, U. M., Veregge, J. R., and Quan, A.G. 1994. Cooperating Expert Systems for the Next Generation of Real-time Monitoring Applications. In *Proc.* 2nd *Internat'l Conference on Expert Systems for Development,* Asian Institute of Technology, Bangkok, Thailand,

Stachowitz, R, A., and Combs, J.B. 1987. Validation of Expert Systems. In *Proc. 20th Annual Hawaii International Conference on System Sciences,* Vol 1, 686-695.

Suwa, M., Scott, A. C., and Shortliffe, E.H. 1982. An Approach To Verifying Completeness and Consistency in a Rule-based Expert System. *Al Magazine,* 3(4) 16-21.